# Sparse Uncertainty Representation in Deep Learning with Inducing Weights

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Bayesian neural networks and deep ensembles represent two modern paradigms of uncertainty quantification in deep learning. Yet these approaches struggle to scale mainly due to memory inefficiency issues, since they require parameter storage several times higher than their deterministic counterparts. To address this, we augment the weight matrix of each layer with a small number of inducing weights, thereby projecting the uncertainty quantification into such low dimensional spaces. We further extend Matheron's conditional Gaussian sampling rule to enable fast weight sampling, which enables our inference method to maintain reasonable runtime as compared with ensembles. Importantly, our approach achieves competitive performance to the state-of-the-art in prediction and uncertainty estimation tasks with fully connected neural networks and ResNets, while reducing the parameter size to $\leq 24.3\%$ of that of a *single* neural network.

## 1 Introduction

Deep learning models are becoming deeper and wider than ever before. From image recognition models such as ResNet-101 (He et al., 2016a) and DenseNet (Huang et al., 2017) to BERT (Xu et al., 2019) and GPT-3 (Brown et al., 2020) for language modelling, deep neural networks have found consistent success in fitting large-scale data. As these models are increasingly deployed in real-world applications, calibrated uncertainty estimates for their predictions become crucial, especially in safety-critical areas such as healthcare. In this regard, Bayesian neural networks (BNNs) (MacKay, 1995; Blundell et al., 2015; Gal & Ghahramani, 2016; Zhang et al., 2020) and deep ensembles (Lakshminarayanan et al., 2017) represent two popular paradigms for estimating uncertainty, which have shown promising results in applications such as (medical) image processing (Kendall & Gal, 2017; Tanno et al., 2017) and out-of-distribution detection (Ovadia et al., 2019).

Though progress has been made, one major obstacle to scaling up BNNs and deep ensembles is the high cost in storing them. Both approaches require the parameter counts to be several times higher than their deterministic counterparts. Although recent efforts have improved memory efficiency (Louizos & Welling, 2017; Swiatkowski et al., 2020; Wen et al., 2020; Dusenberry et al., 2020), they still require storage memory that is higher than storing a deterministic neural network.

Meanwhile, an *infinitely wide* BNN becomes a Gaussian process (GP) that is known for good uncertainty estimates (Neal, 1995; Matthews et al., 2018; Lee et al., 2018). But perhaps surprisingly, this infinitely-wide BNN is "parameter efficient", as its "parameters" are effectively the datapoints, which have a considerably smaller memory footprint than explicitly storing the network weights. To further reduce the computational burden, sparse posterior approximations with a small number of *inducing points* are widely used (Snelson & Ghahramani, 2006; Titsias, 2009), rendering sparse GPs more memory efficient than their neural network counterparts.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Can we bring the advantages of sparse approximations in GPs – which are infinitely-wide neural networks – to finite width deep learning models? We provide an affirmative answer regarding memory efficiency, by proposing an uncertainty quantification framework based on *sparse uncertainty representations*. We present our approach in BNN context, but the proposed approach is also applicable to deep ensembles. In detail, our contributions are as follows:

- We introduce *inducing weights* — an auxiliary variable method with lower dimensional counterparts to the actual weight matrices — for variational inference in BNNs, as well as a memory efficient parameterisation and an extension to ensemble methods (Section 3.1).
- We extend Matheron's rule to facilitate efficient posterior sampling (Section 3.2).
- We show the connection to sparse (deep) GPs, in that inducing weights can be viewed as *projected noisy inducing outputs* in pre-activation output space (Section 3.3).
- We provide an in-depth computation complexity analysis (Section 3.4), showing the significant advantage in terms of parameter efficiency.
- We apply the proposed approach to BNNs and deep ensembles. Experiments in classification, model robustness and out-of-distribution detection tasks show that our inducing weight approaches achieve competitive performance to their counterparts in the original weight space on modern deep architectures for image classification, while reducing the parameter count to $\leq 24.3\%$ of that of a single network.

## 2 Inducing variables for variational inference

Our work is built on variational inference and inducing variables for posterior approximations. Given observations $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ with $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]$, $\mathbf{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_N]$, we would like to fit a neural network $p(\boldsymbol{y}|\boldsymbol{x}, W_{1:L})$ with weights $W_{1:L}$ to the data. BNNs posit a prior distribution $p(W_{1:L})$ over the weights, and construct an approximate posterior $q(W_{1:L})$ to the intractable exact posterior $p(W_{1:L}|\mathcal{D}) \propto p(\mathcal{D}|W_{1:L})p(W_{1:L})$, where $p(\mathcal{D}|W_{1:L}) = p(\mathbf{Y}|\mathbf{X}, W_{1:L}) = \prod_{n=1}^{N} p(\boldsymbol{y}_n|\boldsymbol{x}_n, W_{1:L})$.

**Variational inference** Variational inference (Jordan et al., 1999; Zhang et al., 2018a) constructs an approximation $q(\theta)$ to the posterior $p(\theta|\mathcal{D}) \propto p(\theta)p(\mathcal{D}|\theta)$ by maximising a variational lower-bound:

$$\log p(\mathcal{D}) \geq \mathcal{L}(q(\theta)) := \mathbb{E}_{q(\theta)}\left[\log p(\mathcal{D}|\theta)\right] - \mathbb{KL}\left[q(\theta)||p(\theta)\right]. \tag{1}$$

For BNNs, $\theta = \{W_{1:L}\}$, and a simple choice of $q$ is a fully factorised Gaussian (FFG): $q(W_{1:L}) = \prod_{l=1}^{L}\prod_{i=1}^{d_{out}^l}\prod_{j=1}^{d_{in}^l}\mathcal{N}(m_l^{(i,j)}, v_l^{(i,j)})$, with $m_l^{(i,j)}, v_l^{(i,j)}$ the mean and variance of $W_l^{(i,j)}$ and $d_{in}^l, d_{out}^l$ the respective number of inputs and outputs to layer $l$. The variational parameters are then $\phi = \{m_l^{(i,j)}, v_l^{(i,j)}\}_{l=1}^{L}$. Gradients of $\mathcal{L}$ w.r.t. $\phi$ can be estimated with mini-batches of data (Hoffman et al., 2013) and with Monte Carlo sampling from the $q$ distribution (Titsias & Lázaro-Gredilla, 2014; Kingma & Welling, 2014). By setting $q$ to an FFG, a variational BNN can be trained with similar computational requirements as a deterministic network (Blundell et al., 2015).

**Improved posterior approximation with inducing variables** Auxiliary variable approaches (Agakov & Barber, 2004; Salimans et al., 2015; Ranganath et al., 2016) construct the $q(\theta)$ distribution with an auxiliary variable $a$: $q(\theta) = \int q(\theta|a)q(a)da$, with the hope that a potentially richer mixture distribution $q(\theta)$ can achieve better approximations. As then $q(\theta)$ becomes intractable, an auxiliary variational lower-bound is used to optimise $q(\theta, a)$:

$$\log p(\mathcal{D}) \geq \mathcal{L}(q(\theta, a)) = \mathbb{E}_{q(\theta,a)}[\log p(\mathcal{D}|\theta)] + \mathbb{E}_{q(\theta,a)}\left[\log \frac{p(\theta)r(a|\theta)}{q(\theta|a)q(a)}\right]. \tag{2}$$

Here $r(a|\theta)$ is an auxiliary distribution that needs to be specified, where existing approaches often use a "reverse model" for $r(a|\theta)$. Instead, we define $r(a|\theta)$ in a generative manner: $r(a|\theta)$ is the "posterior" of the following "generative model", whose "evidence" is exactly the prior of $\theta$:

$$r(a|\theta) = \tilde{p}(a|\theta) \propto \tilde{p}(a)\tilde{p}(\theta|a), \quad \text{such that } \tilde{p}(\theta) := \int \tilde{p}(a)\tilde{p}(\theta|a)da = p(\theta). \tag{3}$$

Plugging in (3) to (2):

$$\mathcal{L}(q(\theta, a)) = \mathbb{E}_{q(\theta)}[\log p(\mathcal{D}|\theta)] - \mathbb{E}_{q(a)}\left[\mathbb{KL}[q(\theta|a)||\tilde{p}(\theta|a)]\right] - \mathbb{KL}[q(a)||\tilde{p}(a)]. \tag{4}$$

This approach returns an efficient approximate inference algorithm, translating the complexity of inference in $\theta$ to $a$, if $\dim(a) < \dim(\theta)$ and $q(\theta, a) = q(\theta|a)q(a)$ has the following properties:

1. A "pseudo prior" $\tilde{p}(a)\tilde{p}(\theta|a)$ is defined such that $\int \tilde{p}(a)\tilde{p}(\theta|a)da = p(\theta)$;
2. The conditionals $q(\theta|a)$ and $\tilde{p}(\theta|a)$ are in the same parametric family, so can share parameters;
3. Both sampling $\theta \sim q(\theta)$ and computing $\mathbb{KL}[q(\theta|a)||\tilde{p}(\theta|a)]$ can be done efficiently;
4. The designs of $q(a)$ and $\tilde{p}(a)$ can potentially provide extra advantages (in time and space complexities and/or optimisation easiness).

We call $a$ the *inducing variable* of $\theta$, which is inspired by varationally sparse GP (SVGP) with inducing points (Snelson & Ghahramani, 2006; Titsias, 2009). Indeed SVGP is a special case (see appendix A): $\theta = \mathbf{f}$, $a = \mathbf{u}$, the GP prior is $p(\mathbf{f}|\mathbf{X}) = \mathcal{GP}(\mathbf{0}, \mathbf{K_{XX}})$, $p(\mathbf{u}) = \mathcal{GP}(\mathbf{0}, \mathbf{K_{ZZ}})$, $\tilde{p}(\mathbf{f}, \mathbf{u}) = p(\mathbf{u})p(\mathbf{f}|\mathbf{X}, \mathbf{u})$, $q(\mathbf{f}|\mathbf{u}) = p(\mathbf{f}|\mathbf{X}, \mathbf{u})$, $q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{X}, \mathbf{u})q(\mathbf{u})$, and $\mathbf{Z}$ are the optimisable inducing inputs. The variational lower-bound is $\mathcal{L}(q(\mathbf{f}, \mathbf{u})) = \mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{Y}|\mathbf{f})] - \mathbb{KL}[q(\mathbf{u})||p(\mathbf{u})]$, and the variational parameters are $\phi = \{\mathbf{Z}, \text{distribution parameters of } q(\mathbf{u})\}$. SVGP satisfies the marginalisation constraint (3) by definition, and it has $\mathbb{KL}[q(\mathbf{f}|\mathbf{u})||\tilde{p}(\mathbf{f}|\mathbf{u})] = 0$. Also by using small $M = \dim(\mathbf{u})$ and exploiting the $q$ distribution design, SVGP reduces run-time from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$ where $N$ is the number of inputs in $\mathbf{X}$, meanwhile it also makes storing a full Gaussian $q(\mathbf{u})$ affordable. Lastly, $\mathbf{u}$ can be whitened, leading to the "pseudo prior" $\tilde{p}(\mathbf{f}, \mathbf{v}) = p(\mathbf{f}|\mathbf{X}, \mathbf{u} = \mathbf{K}_{\mathbf{ZZ}}^{1/2}\mathbf{v})\tilde{p}(\mathbf{v}), \tilde{p}(\mathbf{v}) = \mathcal{N}(\mathbf{v}; \mathbf{0}, \mathbf{I})$ which could bring potential benefits in optimisation.

We emphasise that the introduction of "pseudo prior" does *not* change the *probabilistic model* as long as the marginalisation constraint (3) is satisfied. In the rest of the paper we assume the constraint (3) holds and write $p(\theta, a) := \tilde{p}(\theta, a)$. It might seem unclear how to design such $\tilde{p}(\theta, a)$ for an arbitrary probabilistic model, however, for a Gaussian prior on $\theta$ the rules for computing conditional Gaussian distributions can be used to construct $\tilde{p}$. In section 3 we exploit these rules to develop an efficient approximate inference method for Bayesian neural networks with inducing weights.

# 3 Sparse uncertainty representation with inducing weights

## 3.1 Inducing weights for neural network parameters

Following the above design principles, we introduce to each network layer $l$ a *smaller* inducing weight matrix $U_l$ to assist approximate posterior inference in $W_l$. Therefore in our context, $\theta = W_{1:L}$ and $a = U_{1:L}$. In the rest of the paper, we assume a factorised prior across layers $p(W_{1:L}) = \prod_l p(W_l)$, and drop the $l$ indices when the context is clear to ease notation.

**Augmenting network layers with inducing weights** Suppose the weight $W \in \mathbb{R}^{d_{out} \times d_{in}}$ has a Gaussian prior $p(W) = p(\text{vec}(W)) = \mathcal{N}(0, \sigma^2 I)$ where $\text{vec}(W)$ concatenates the columns of the weight matrix into a vector. A first attempt to augment $p(\text{vec}(W))$ with an inducing weight variable $U \in \mathbb{R}^{M_{out} \times M_{in}}$ may be to construct a multivariate Gaussian $p(\text{vec}(W), \text{vec}(U))$, such that $\int p(\text{vec}(W), \text{vec}(U))dU = \mathcal{N}(0, \sigma^2 I)$. This means for the joint covariance matrix of $(\text{vec}(W), \text{vec}(U))$, it requires the block corresponding to the covariance of $\text{vec}(W)$ to match the prior covariance $\sigma^2 I$. We are then free to parameterise the rest of the entries in the joint covariance matrix, as long as this full matrix remains positive definite. Now the conditional distribution $p(W|U)$ is a function of these parameters, and the conditional sampling from $p(W|U)$ is further discussed in Appendix B.1. Unfortunately, as $\dim(\text{vec}(W))$ is typically large (e.g. of the order of $10^7$), using a full covariance Gaussian for $p(\text{vec}(W), \text{vec}(U))$ becomes computationally intractable.

We address this issue with matrix normal distributions (Gupta & Nagar, 2018). The prior $p(\text{vec}(W)) = \mathcal{N}(\mathbf{0}, \sigma^2 I)$ has an equivalent matrix normal distribution form as $p(W) = \mathcal{MN}(0, \sigma_r^2 I, \sigma_c^2 I)$, with $\sigma_r, \sigma_c > 0$ the row and column standard deviations satisfying $\sigma = \sigma_r \sigma_c$. Now we introduce the inducing variable $U$ in matrix space, as well as two auxiliary variables $U_r \in \mathbb{R}^{M_{out} \times d_{in}}$, $U_c \in \mathbb{R}^{d_{out} \times M_{in}}$, so that the full augmented prior is:

$$\begin{pmatrix} W & U_c \\ U_r & U \end{pmatrix} \sim p(W, U_c, U_r, U) := \mathcal{MN}(0, \Sigma_r, \Sigma_c), \tag{5}$$

$$\text{with} \quad L_r = \begin{pmatrix} \sigma_r I & 0 \\ Z_r & D_r \end{pmatrix} \quad \text{s.t.} \quad \Sigma_r = L_r L_r^\top = \begin{pmatrix} \sigma_r^2 I & \sigma_r Z_r^\top \\ \sigma_r Z_r & Z_r Z_r^\top + D_r^2 \end{pmatrix}$$

$$\text{and} \quad L_c = \begin{pmatrix} \sigma_c I & 0 \\ Z_c & D_c \end{pmatrix} \quad \text{s.t.} \quad \Sigma_c = L_c L_c^\top = \begin{pmatrix} \sigma_c^2 I & \sigma_c Z_c^\top \\ \sigma_c Z_c & Z_c Z_c^\top + D_c^2 \end{pmatrix}.$$

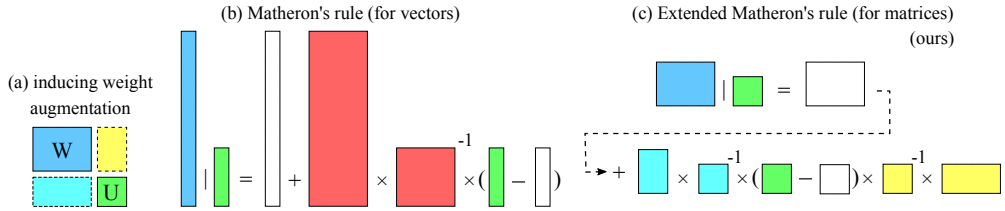



Figure 1: Visualisation of (a) the inducing weight augmentation, and compare (b) the original Matheron's rule to (c) our extended version. White blocks represent samples from the joint Gaussian.

See Figure 1(a) for a visualisation. Matrix normal distributions have similar marginalization and conditioning properties as multivariate Gaussian distributions. Therefore the marginalisation constraint (3) is satisfied for any $Z_c \in \mathbb{R}^{M_{in} \times d_{in}}$, $Z_r \in \mathbb{R}^{M_{out} \times d_{out}}$ and diagonal matrices $D_c, D_r$. The marginal distribution of $U$ is $p(U) = \mathcal{MN}(0, \Psi_r, \Psi_c)$ with $\Psi_r = Z_r Z_r^\top + D_r^2$ and $\Psi_c = Z_c Z_c^\top + D_c^2$. In the experiments we use *whitened* inducing weights which transforms $U$ so that $p(U) = \mathcal{MN}(0, I, I)$ (Appendix F), but for clarity we continue with the above formulas in the main text.

The matrix normal parameterisation introduces two additional variables $U_r, U_c$ without providing additional expressiveness. Hence it is desirable to integrate them out, leading to a joint multivariate normal with Khatri-Rao product structure for the covariance:

$$p(\mathrm{vec}(W), \mathrm{vec}(U)) = \mathcal{N}\left(0, \begin{pmatrix} \sigma_c^2 I \otimes \sigma_r^2 I & \sigma_c Z_c^\top \otimes \sigma_r Z_r^\top \\ \sigma_c Z_c \otimes \sigma_r Z_r & \Psi_c \otimes \Psi_r \end{pmatrix}\right). \tag{6}$$

As the dominating memory complexity here is $\mathcal{O}(d_{out} M_{out} + d_{in} M_{in})$ which comes from storing $Z_r$ and $Z_c$, we see that the matrix normal parameterisation of the augmented prior is memory efficient.

**Posterior approximation in the joint space** We construct a factorised posterior approximation across the layers: $q(W_{1:L}, U_{1:L}) = \prod_l q(W_l|U_l) q(U_l)$. Below we discuss options for $q(W|U)$.

The simplest option is $q(W|U) = p(\mathrm{vec}(W)|\,\mathrm{vec}(U)) = \mathcal{N}(\mu_{W|U}, \Sigma_{W|U})$, similar to sparse GPs. A slightly more flexible variant adds a rescaling term $\lambda^2$ to the covariance matrix, which allows efficient KL computation (Appendix C):

$$q(W|U) = q(\mathrm{vec}(W)|\,\mathrm{vec}(U)) = \mathcal{N}(\mu_{W|U}, \lambda^2 \Sigma_{W|U}), \tag{7}$$

$$R(\lambda) := \mathbb{KL}\left[q(W|U)||p(W|U)\right] = d_{in} d_{out}(0.5\lambda^2 - \log\lambda - 0.5). \tag{8}$$

Plugging $\theta = W_{1:L}, a = U_{1:L}$ and (8) into (4) results in the following variational lower-bound

$$\mathcal{L}(q(W_{1:L}, U_{1:L})) = \mathbb{E}_{q(W_{1:L})}[\log p(\mathcal{D}|W_{1:L})] - \textstyle\sum_{l=1}^{L}(R(\lambda_l) + \mathbb{KL}[q(U_l)||p(U_l)]), \tag{9}$$

with $\lambda_l$ the associated scaling parameter for $q(W_l|U_l)$. Again as the choices of $Z_c, Z_r, D_c, D_r$ do not change the marginal prior $p(W)$, we are safe to optimise them as well. Therefore the variational parameters are now $\phi = \{Z_c, Z_r, D_c, D_r, \lambda, \text{dist. params. of } q(U)\}$ for each layer.

**Two choices of $q(U)$** A simple choice is FFG $q(\mathrm{vec}(U)) = \mathcal{N}(\boldsymbol{m}_u, \mathrm{diag}(\boldsymbol{v}_u))$, which performs mean-field inference in $U$ space (c.f. Blundell et al., 2015), and here $\mathbb{KL}[q(U)||p(U)]$ has a closed-form solution. Another choice is a "mixture of delta measures" $q(U) = \frac{1}{K}\sum_{k=1}^{K} \delta(U = U^{(k)})$, i.e. we keep $K$ distinct sets of parameters $\{U_{1:L}^{(k)}\}_{k=1}^{K}$ in inducing space that are projected back into the original parameter space via the *shared* conditionals $q(W_l|U_l)$ to obtain the weights. This approach can be viewed as constructing "deep ensembles" in $U$ space, and we follow ensemble methods (e.g. Lakshminarayanan et al., 2017) to drop $\mathbb{KL}[q(U)||p(U)]$ in (9).

Often $U$ is chosen to have significantly lower dimensions than $W$, i.e. $M_{in} << d_{in}$ and $M_{out} << d_{out}$. As $q(W|U)$ and $p(W|U)$ only differ in the covariance scaling constant, $U$ can be regarded as a *sparse representation of uncertainty* for the network layer, as the major updates in (approximate) posterior belief is quantified by $q(U)$.

### 3.2 Efficient sampling with extended Matheron's rule

Computing the variational lower-bound (9) requires samples from $q(W)$, which requires an efficient sampling procedure for $q(W|U)$. Unfortunately, $q(W|U)$ derived from (6) & (7) is not a matrix

normal, so *direct* sampling is prohibitive. To address this challenge, we extend Matheron's rule (Journel & Huijbregts, 1978; Hoffman & Ribak, 1991; Doucet, 2010) to efficiently sample from $q(W|U)$ (derivations in Appendix D). The idea is that one can sample from a conditional Gaussian $W \sim q(W|U)$ by transforming a sample from a joint matrix normal distribution:

$$W = \lambda \bar{W} + \sigma Z_r^\top \Psi_r^{-1}(U - \lambda \bar{U})\Psi_c^{-1} Z_c; \qquad \bar{W}, \bar{U} \sim p(\bar{W}, \bar{U}_c, \bar{U}_r, \bar{U}) = \mathcal{MN}(0, \Sigma_r, \Sigma_c). \quad (10)$$

Here $\bar{W}, \bar{U} \sim p(\bar{W}, \bar{U}_c, \bar{U}_r, \bar{U})$ means we first sample $\bar{W}, \bar{U}_c, \bar{U}_r, \bar{U}$ from the joint then drop $\bar{U}_c, \bar{U}_r$; in fact $\bar{U}_c, \bar{U}_r$ are never computed. Details are in Appendix D, in short $\bar{W}, \bar{U}$ can be obtained by:

$$\begin{aligned} &\bar{W} = \sigma E_1,\ \bar{U} = Z_r E_1 Z_c^\top + \hat{L}_r \tilde{E}_2 D_c + D_r \tilde{E}_3 \hat{L}_c^\top + D_r E_4 D_c, \\ &E_1 \sim \mathcal{MN}(0, I_{d_{out}}, I_{d_{in}}); \quad \tilde{E}_2, \tilde{E}_3, E_4 \sim \mathcal{MN}(0, I_{M_{out}}, I_{M_{in}}), \\ &\hat{L}_r = \text{chol}(Z_r Z_r^\top),\ \hat{L}_c = \text{chol}(Z_c Z_c^\top). \end{aligned} \quad (11)$$

The run-time cost is $\mathcal{O}(2M_{out}^3 + 2M_{in}^3 + d_{out}M_{out}M_{in} + M_{in}d_{out}d_{in})$ required by inverting $\Psi_r, \Psi_c$, computing $\hat{L}_r$, $\hat{L}_c$, and the matrix products. The extended Matheron's rule is visualised in Figure 1 with a comparison to the original Matheron's rule for sampling from $q(\text{vec}(W)|\,\text{vec}(U))$. Note that the original rule requires joint sampling from (6) (i.e. sampling the white blocks in Figure 1(b)) which has $\mathcal{O}((d_{out}d_{in} + M_{out}M_{in})^3)$ cost. Therefore our recipe avoids inverting and multiplying big matrices, resulting in a significant speed-up for conditional sampling.

### 3.3 A function-space perspective on inducing weights

We present the proposed approach again but from a function-space inference perspective. Assume a network layer computes the following transformation of the input $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N], \boldsymbol{x}_i \in \mathbb{R}^{d_{in} \times 1}$: $\mathbf{F} = W\mathbf{X},\ \mathbf{H} = g(\mathbf{F})$, where $W \in \mathbb{R}^{d_{out} \times d_{in}}$ and $g(\cdot)$ is the non-linearity. As $W$ has a Gaussian prior $p(\text{vec}(W)) = \mathcal{N}(0, \sigma^2 I)$, each of the rows in $\mathbf{F} = [\mathbf{f}_1, ..., \mathbf{f}_{d_{out}}]^\top, \mathbf{f}_i \in \mathbb{R}^{N \times 1}$ has a Gaussian process form with linear kernel: $\mathbf{f}_i|\mathbf{X} \sim \mathcal{GP}(\mathbf{0}, \mathbf{K}_{\mathbf{XX}}), \mathbf{K}_{\mathbf{XX}}(m, n) = \sigma^2 \boldsymbol{x}_m^\top \boldsymbol{x}_n$. Inference on $\mathbf{F}$ directly has $\mathcal{O}(N^3 + d_{out}N^2)$ cost, so a sparse approximation is needed. Slightly different from the usual approach, we introduce auxiliary variables $U_c = [\mathbf{u}_1^c, ..., \mathbf{u}_{d_{out}}^c]^\top \in \mathbb{R}^{d_{out} \times M_{in}}$ as follows, using shared "inducing inputs" $Z_c^\top \in \mathbb{R}^{d_{in} \times M_{in}}$:

$$p(\mathbf{f}_i, \hat{\mathbf{u}}_i|\mathbf{X}) = \mathcal{GP}\left(\mathbf{0}, \mathbf{K}_{[\mathbf{X}, Z_c^\top], [\mathbf{X}, Z_c^\top]}\right), \quad p(\mathbf{u}_i^c|\hat{\mathbf{u}}_i) = \mathcal{N}\left(\hat{\mathbf{u}}_i/\sigma_c, \sigma_r^2 D_c^2\right), \quad (12)$$

By marginalising out the "noiseless inducing outputs" $\{\hat{\mathbf{u}}_i\}$ (details in Appendix E), we can compute the marginal distributions as $p(U_c) := p(\{\mathbf{u}_i^c\}) = \mathcal{MN}(\mathbf{0}, \sigma_r^2 I, \Psi_c)$ and

$$p(\mathbf{F}|\mathbf{X}, U_c) = \mathcal{MN}\left(U_c \Psi_c^{-1} \sigma_c Z_c \mathbf{X}, \sigma_r^2 I, \mathbf{X}^\top \sigma_c^2 (I - Z_c^\top \Psi_c^{-1} Z_c)\mathbf{X}\right). \quad (13)$$

Note $p(W|U_c) = \mathcal{MN}\left(U_c \Psi_c^{-1} \sigma_c Z_c, \sigma_r^2 I, \sigma_c^2 (I - Z_c^\top \Psi_c^{-1} Z_c)\right)$ (Appendix B.2). Since $W \sim \mathcal{MN}(\mathbf{M}, \Sigma_1, \Sigma_2)$ leads to $W\mathbf{X} \overset{d}{\sim} \mathcal{MN}(\mathbf{MX}, \Sigma_1, \mathbf{X}^\top \Sigma_2 \mathbf{X})$, this means $p(\mathbf{F}|\mathbf{X}, U_c)$ is the push-forward distribution of $p(W|U_c)$ for $\mathbf{F} = W\mathbf{X}$:

$$\mathbf{F} \sim p(\mathbf{F}|\mathbf{X}, U_c) \quad \Leftrightarrow \quad W \sim p(W|U_c),\ \mathbf{F} = W\mathbf{X}.$$

As $\{\mathbf{u}_i^c\}$ are the "noisy" versions of $\{\hat{\mathbf{u}}_i\}$ in $\mathbf{f}$ space, $U_c$ can be viewed as "scaled noisy inducing outputs" in function space (see the red bars in the 2nd row of Figure 2).

So far the inducing variables $U_c$ is used as a compact representation of the correlation *between* columns of $\mathbf{F}$ only. In other words the output dimension $d_{out}$ for each $\mathbf{f}_i$ (and $\mathbf{u}_i^c$) remains large (e.g. $> 1000$ in a fully connected layer). Therefore dimensionality reduction can be applied to the *column vectors* of $U_c$ and $\mathbf{F}$. In Appendix E we present a generative approach to do so by extending probabilistic PCA (Tipping & Bishop, 1999) to matrix normals: $p(U_c) = \int p(U_c|U)p(U)dU$, where the projection's parameters are $\{Z_r, D_r\}$, and $p(U_c, U)$ matches the marginals of (5). This means $U$ can be viewed as the "noisy projected inducing output" of the GP whose corresponding "inducing inputs" are $Z_c^\top$ (see the red bars in Figure 2). Similarly column vectors in $U_r\mathbf{X}$ can be viewed as noisy projections of column vectors in $\mathbf{F}$.

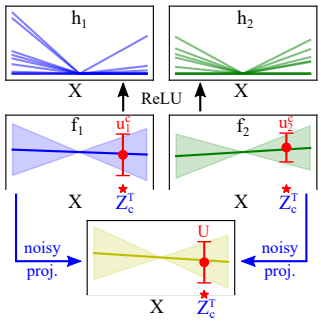



Figure 2: Visualising $U$ variables in pre-activation spaces.

Table 1: Computational complexity per layer. We assume $W \in \mathbb{R}^{d_{out} \times d_{in}}$, $U \in \mathbb{R}^{M_{out} \times M_{in}}$, and $K$ forward passes for each of the $N$ inputs. ($^*$ uses a parallel computing friendly vectorisation technique (Wen et al., 2020) for further speed-up.)

| Method | Time complexity | Storage complexity |
|---|---|---|
| Deterministic-$W$ | $\mathcal{O}(N d_{in} d_{out})$ | $\mathcal{O}(d_{in} d_{out})$ |
| FFG-$W$ | $\mathcal{O}(NK d_{in} d_{out})$ | $\mathcal{O}(2 d_{in} d_{out})$ |
| Ensemble-$W$ | $\mathcal{O}(NK d_{in} d_{out})$ | $\mathcal{O}(K d_{in} d_{out})$ |
| Matrix-normal-$W$ | $\mathcal{O}(NK d_{in} d_{out})$ | $\mathcal{O}(d_{in} d_{out} + d_{in} + d_{out})$ |
| $k$-tied FFG-$W$ | $\mathcal{O}(NK d_{in} d_{out})$ | $\mathcal{O}(d_{in} d_{out} + k(d_{in} + d_{out}))$ |
| rank-1 BNN | $\mathcal{O}(NK d_{in} d_{out})^*$ | $\mathcal{O}(d_{in} d_{out} + 2(d_{in} + d_{out}))$ |
| FFG-$U$ | $\mathcal{O}(NK d_{in} d_{out} + 2M_{in}^3 + 2M_{out}^3 + K(d_{out} M_{out} M_{in} + M_{in} d_{out} d_{in}))$ | $\mathcal{O}(d_{in} M_{in} + d_{out} M_{out} + 2 M_{in} M_{out})$ |
| Ensemble-$U$ | same as above | $\mathcal{O}(d_{in} M_{in} + d_{out} M_{out} + K M_{in} M_{out})$ |

In Appendix E we further show that the push-forward distribution $q(\mathbf{F}|\mathbf{X}, U)$ of $q(W|U)$ only differs from $p(\mathbf{F}|\mathbf{X}, U)$ in the covariance matrices up to the scaling constant $\lambda$. Therefore the resulting function-space variational objective is almost identical to (9), except for scaling coefficients on the $R(\lambda_l)$ terms to account for the change in dimensionality from $\mathrm{vec}(W)$ to $\mathrm{vec}(\mathbf{F})$. This result nicely connects posterior inference in weight- and function-space.

### 3.4 Computational complexities

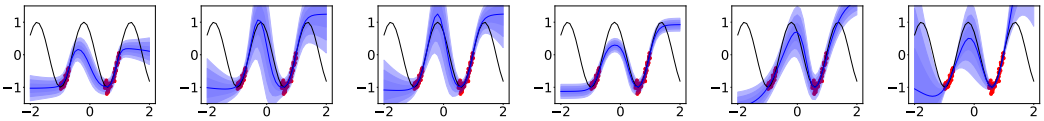
In Table 1 we report the computational complexity figures for two types of inducing weight approaches: FFG $q(U)$ (FFG-$U$) and Delta mixture $q(U)$ (Ensemble-$U$). Baseline approaches include: Deterministic-$W$, variational inference with FFG $q(W)$ (FFG-$W$, Blundell et al., 2015), deep ensemble in $W$ (Ensemble-$W$, Lakshminarayanan et al., 2017), as well as parameter efficient approaches such as matrix-normal $q(W)$ (Matrix-normal-$W$, Louizos & Welling (2017)), variational inference with $k$-tied FFG $q(W)$ ($k$-tied FFG-$W$, Swiatkowski et al. (2020)), and rank-1 BNN (Dusenberry et al., 2020). The gain in memory is significant for the inducing weight approaches, in fact with $M_{in} < d_{in}$ and $M_{out} < d_{out}$ the parameter storage requirement is smaller than a single deterministic neural network. The major overhead in run-time comes from the extended Matheron's rule for sampling $q(W|U)$. Some of the computations there are performed only once, and in our experiments we show that by using a relatively low-dimensional $U$ and large batch-sizes, the overhead is acceptable.

## 4 Experiments

We evaluate the inducing weight approaches on regression, classification and related uncertainty estimation tasks. The goal is to demonstrate competitive performance to popular $W$-space uncertainty estimation methods while using significantly fewer parameters. We acknowledge that existing parameter efficient approaches for uncertainty estimation (e.g. k-tied or rank-1 BNNs) have achieved close performance to deep ensembles. However, *none* of them reduces the parameter count to be smaller than that of a *single* network. Therefore we decide *not* to include these baselines and instead focus on comparing: (1) variational inference with FFG $q(W)$ (FFG-$W$, Blundell et al., 2015) v.s. FFG $q(U)$ (FFG-$U$, ours); (2) deep ensemble in $W$ space (Ensemble-$W$, Lakshminarayanan et al., 2017) v.s. ensemble in $U$ space (Ensemble-$U$, ours). Another baseline is training a deterministic neural network with maximum likelihood. Details and additional results are in Appendix G & H.

### 4.1 Synthetic 1-D regression

The synthetic regression task follows Foong et al. (2019), which has two clusters of inputs $x_1 \sim \mathcal{U}[-1, -0.7]$, $x_2 \sim \mathcal{U}[0.5, 1]$, and targets $y \sim \mathcal{N}(\cos(4x + 0.8), 0.01)$. For reference we show the exact posterior results using the NUTS sampler (Hoffman & Gelman, 2014). The results are visualised in Figure 3 with predictive mean in blue, and up to three standard deviations as shaded area. Similar to historical results, FFG-$W$ fails to represent the increased uncertainty away from the data and in between clusters. While underestimating predictive uncertainty overall, FFG-$U$ shows a small increase in predictive uncertainty away from the data. In contrast, a per-layer full covariance Gaussian in both weight (FCG-$W$) and inducing space (FCG-$U$) as well as Ensemble-$U$ better capture the increased predictive variance, although the mean function is more similar to that of FFG-$W$.



(a) FFG-$U$ (b) FCG-$U$ (c) Ensemble-$U$ (d) FFG-$W$ (e) FCG-$W$ (f) NUTS

Figure 3: Toy regression results, with observations in red dots and the ground truth function in black.

Table 2: CIFAR in-distribution metrics (in %).

| Method | CIFAR10 Acc. ↑ | CIFAR10 ECE ↓ | CIFAR100 Acc. ↑ | CIFAR100 ECE ↓ |
|---|---|---|---|---|
| Deterministic | 94.72 | 4.46 | 75.73 | 19.69 |
| Ensemble-W | 95.90 | 1.08 | 79.33 | 6.51 |
| FFG-W | 94.13 | 0.50 | 74.44 | 4.24 |
| FFG-U | 94.40 | 0.64 | 75.37 | 2.29 |
| Ensemble-U | 94.94 | 0.45 | 75.97 | 1.12 |

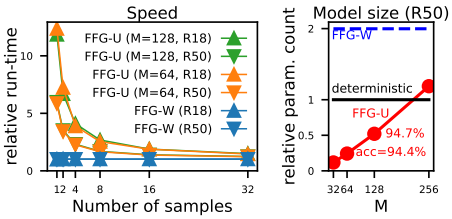



Figure 4: Resnet run-times & model sizes.

### 4.2 Classification and in-distribution calibration

As the core empirical evaluation, we train Resnet-50 models (He et al., 2016b) on CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009). To avoid underfitting issues with FFG-W, a useful trick is to set an upper limit $\sigma^2_{max}$ on the variance of $q(W)$ (Louizos & Welling, 2017). This trick is similarly applied to the $U$-space methods, where we cap $\lambda \leq \lambda_{max}$ for $q(W|U)$, and for FFG-$U$ we also set $\sigma^2_{max}$ for the variance of $q(U)$. In convolution layers, we treat the 4D weight tensor $W$ of shape $(c_{out}, c_{in}, h, w)$ as a $c_{out} \times c_{in}hw$ matrix. We use $U$ matrices of shape $64 \times 64$ for all layers (i.e. $M = M_{in} = M_{out} = 64$), except that for CIFAR-10 we set $M_{out} = 10$ for the last layer.

In Table 2 we report test accuracy and test expected calibration error (ECE, Guo et al., 2017) as a first evaluation of the uncertainty estimates. Overall, Ensemble-$W$ achieves the highest accuracy, but is not as well-calibrated as variational methods. For the inducing weight approaches, Ensemble-$U$ outperforms FFG-$U$ on both datasets; overall it performs the best on the more challenging CIFAR-100 dataset (close-to-Ensemble-$W$ accuracy and lowest ECE). Tables 4 & 5 in the Appendix show that increasing the $U$ dimensions to $M = 128$ improves accuracy but leads to slightly worse calibration.

In Figure 4 we show prediction run-times for batch-size $= 500$ on an NVIDIA Tesla V100 GPU, relative to those of an ensemble of deterministic nets, as well as relative parameter sizes to a single ResNet-50. The extra run-times for the inducing methods come from computing the extended Matheron's rule. However, as they can be calculated once and cached for drawing multiple samples, the overhead reduces to a small factor when using larger number of samples $K$, especially for the bigger Resnet-50. More importantly, when compared to a *deterministic* ResNet-50, the inducing weight models reduce the parameter count by over 75% $(5,710,902$ vs. $23,520,842)$ for $M = 64$.

**Hyper-parameter choices** We visualise in Figure 5 the accuracy and ECE results for computationally lighter inducing weight ResNet-18 models with different hyper-parameters. Performance in both metrics improves as the $U$ matrix size $M$ is increased (right-most panels), and the results for $M = 64$ and $M = 128$ are fairly similar. Also setting proper values for $\lambda_{max}, \sigma_{max}$ is key

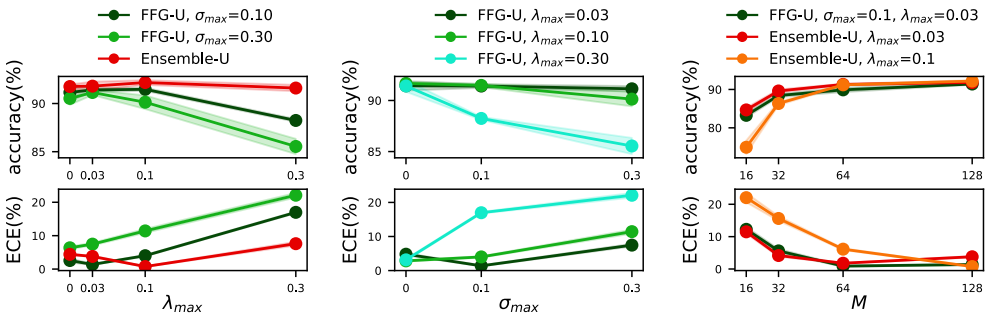



Figure 5: Ablation study: average CIFAR-10 accuracy (↑) and ECE (↓) for the inducing weight methods on ResNet-18. In the first two columns $M = 128$ for $U$ dimensions. For $\lambda_{max}, \sigma_{max} = 0$ we use point estimates for $U, W$ respectively.

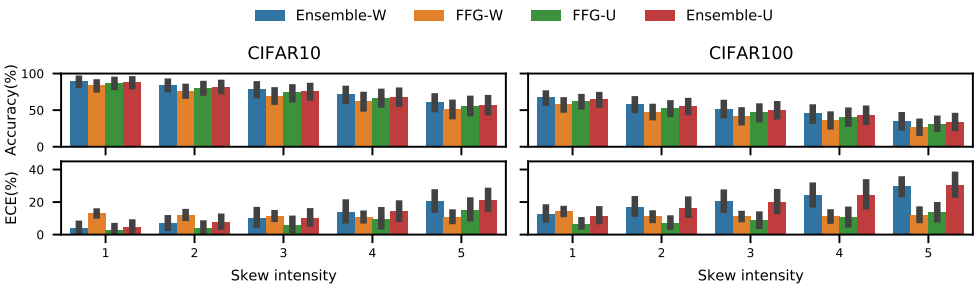



Figure 6: Mean±two errs. for Acc↑ and ECE↓ on corrupted CIFAR (Hendrycks & Dietterich, 2019).

Table 3: OOD detection metrics for Resnet-50 trained on CIFAR10/100.

| In-dist → OOD | C10 → C100 | | C10 → SVHN | | C100 → C10 | | C100 → SVHN | |
|---|---|---|---|---|---|---|---|---|
| Method / Metric | AUROC | AUPR | AUROC | AUPR | AUROC | AUPR | AUROC | AUPR |
| Deterministic | .84±.00 | .80±.00 | .93±.01 | .85±.01 | .74±.00 | .74±.00 | .81±.01 | .72±.02 |
| Ensemble-W | .89 | .89 | .95 | .92 | .78 | .79 | .86 | .78 |
| FFG-W | .88±.00 | .90±.00 | .90±.01 | .86±.01 | .76±.00 | .79±.00 | .80±.01 | .69±.01 |
| FFG-U | .89±.00 | .91±.00 | .94±.01 | .91±.01 | .77±.00 | .79±.00 | .83±.01 | .74±.01 |
| Ensemble-U | .90±.00 | .91±.00 | .93±.00 | .91±.00 | .77±.00 | .79±.00 | .82±.01 | .72±.02 |

to the improved results. The left-most panels show that with fixed $\sigma_{max}$ values (or Ensemble-$U$), the preferred conditional variance cap values $\lambda_{max}$ are fairly small (but still larger than 0 which corresponds to a point estimate for $W$ given $U$). For $\sigma_{max}$ which controls variance in $U$ space, we see from the top middle panel that the accuracy metric is fairly robust to $\sigma_{max}$ as long as $\lambda_{max}$ is not too large. But for ECE, a careful selection of $\sigma_{max}$ is required (bottom middle panel).

### 4.3 Robustness, out-of-distribution detection and pruning

To investigate the models' robustness to distribution shift, we compute predictions on corrupted CIFAR datasets (Hendrycks & Dietterich, 2019) after training on clean data. Figure 6 shows accuracy and ECE results for the ResNet-50 models. Ensemble-$W$ is the most accurate model across skew intensities, while FFG-$W$, though performing well on clean data, returns the worst accuracy under perturbation. The inducing weight methods perform competitively to Ensemble-$W$ with Ensemble-$U$ being slightly more accurate than FFG-$U$ as on the clean data. For ECE, FFG-$U$ outperforms Ensemble-$U$ and Ensemble-$W$, which are similarly calibrated. Interestingly, while the accuracy of FFG-$W$ decays quickly as the data is perturbed more strongly, its ECE remains roughly constant.

Table 3 further presents the utility of the maximum predicted probability for out-of-distribution (OOD) detection. The metrics are the area under the receiver operator characteristic (AUROC) and the precision-recall curve (AUPR). The inducing-weight methods perform similarly to Ensemble-$W$; all three outperform FFG-$W$ and deterministic networks across the board.

**Parameter pruning** We further investigate pruning as a pragmatic alternative for more parameter-efficient inference. For FFG-$U$, we prune entries of the $Z$ matrices, which contribute the largest number of parameters to the inducing methods, with the smallest magnitude. For FFG-$W$ we follow Graves (2011) in setting different fractions of $W$ to 0 depending on their variational mean-to-variance ratio and repeat the previous experiments after fine-tuning the distributions on the remaining variables. We stress that, unlike FFG-$U$, the FFG-$W$ pruning corresponds to a post-hoc change of the probabilistic model and no longer performs inference in the original weight-space.

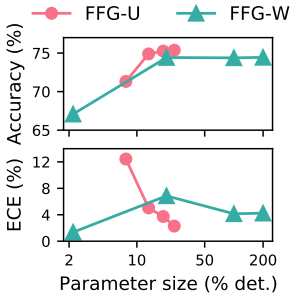



Figure 7: CIFAR100 pruning accuracy(↑) & ECE(↓). Right-most points are w/out pruning.

For FFG-$W$, pruning 90% of the parameters (leaving 20% of parameters as compared to its deterministic counterpart) worsens the ECE, in particular on CIFAR100, see Figure 7. Further pruning to 1% worsens the accuracy and the OOD detection results as well. On the other hand, pruning 50% of the $Z$ matrices for FFG-$U$ reduces the parameter count to 13.2% of a deterministic net, at the cost of only slightly worse calibration. See Appendix H for the full results.

## 5 Related Work

**Parameter-efficient uncertainty quantification methods** Recent research has proposed Gaussian posterior approximations for BNNs with efficient covariance structure (Ritter et al., 2018; Zhang et al., 2018b; Mishkin et al., 2018). The inducing weight approach differs from these in introducing structure via a hierarchical posterior with low-dimensional auxiliary variables. Another line of work reduces the memory overhead via efficient parameter sharing (Louizos & Welling, 2017; Wen et al., 2020; Swiatkowski et al., 2020; Dusenberry et al., 2020). The third category of work considers a hybrid approach, where only a selective part of the neural network receives Bayesian treatments, and the other weights remain deterministic (Bradshaw et al., 2017; Daxberger et al., 2020). However, both types of approaches maintain a "mean parameter" for the weights, making the memory footprint at least that of storing a deterministic neural network. Instead, our approach shares parameters via the augmented prior with efficient low-rank structure, *reducing* the memory use compared to a deterministic network. In a similar spirit, Izmailov et al. (2019) perform inference in a $d$-dimensional sub-space obtained from PCA on weights collected from an SGD trajectory. But this approach does not leverage the layer-structure of neural networks and requires $d\times$ memory of a single network.

As a side note, mean-field VI approaches have shown success in network pruning but only in terms of maintaining a minimum accuracy level (Graves, 2011; Louizos et al., 2017; Havasi et al., 2019). To the best of our knowledge, our empirical study presents the first evaluation for pruning methods in maintaining uncertainty estimation quality.

**Sparse GP and function-space inference** As BNNs and GPs are closely related (Neal, 1995; Matthews et al., 2018; Lee et al., 2018), recent efforts have introduced GP-inspired techniques to BNNs (Ma et al., 2019; Sun et al., 2019; Khan et al., 2019; Ober & Aitchison, 2020). Compared to weight-space inference, function-space inference is appealing as its uncertainty is more directly relevant for predictive uncertainty estimation. While the inducing weight approach performs computations in weight-space, Section 3.3 establishes the connection to function-space posteriors. Our approach is related to sparse deep GP methods with $U_c$ having similar interpretations as inducing outputs in e.g. Salimbeni & Deisenroth (2017). The major difference is that $U$ lies in a low-dimensional space, projected from the pre-activation output space of a network layer.

**Priors on neural network weights** Hierarchical priors for weights has also been explored (Louizos et al., 2017; Krueger et al., 2017; Atanov et al., 2019; Ghosh et al., 2019; Karaletsos & Bui, 2020). However, we emphasise that $\tilde{p}(W, U)$ is a *pseudo prior* that is constructed to *assist posterior inference* rather than to *improve model design*. Indeed, parameters associated with the inducing weights are optimisable for improving posterior approximations. Our approach can be adapted to other priors, e.g. for a Horseshoe prior $p(\theta, \nu) = p(\theta|\nu)p(\nu) = \mathcal{N}(\theta; 0, \nu^2)C^+(\nu; 0, 1)$, the pseudo prior can be defined as $\tilde{p}(\theta, \nu, a) = \tilde{p}(\theta|\nu, a)\tilde{p}(a)p(\nu)$ such that $\int \tilde{p}(\theta|\nu, a)\tilde{p}(a)da = p(\theta|\nu)$. In general, pseudo priors have found broader success in Bayesian computation (Carlin & Chib, 1995).

## 6 Conclusion

We have proposed a parameter-efficient uncertainty quantification framework for neural networks. It augments each of the network layer weights with a small matrix of inducing weights, and by extending Matheron's rule to matrix-normal related distributions, maintains a relatively small run-time overhead compared with ensemble methods. Critically, experiments on prediction and uncertainty estimation tasks show the competence of the inducing weight methods to the state-of-the-art, while reducing the parameter count to under a quarter of a deterministic ResNet-50 before pruning. This represents a significant improvement over prior Bayesian and deep ensemble techniques, which so far have not managed to go below this threshold despite various attempts of matching it closely.

Several directions are to be explored in the future. First, modelling correlations across layers might further improve the inference quality. We outline an initial approach leveraging inducing variables in Appendix F. Second, based on the function-space interpretation of inducing weights, better initialisation techniques can be inspired from the sparse GP and dimension reduction literature. Similarly, this interpretation might suggest other innovative pruning approaches for the inducing weight method, thereby achieving further memory savings. Lastly, the small run-time overhead of our approach can be mitigated by a better design of the inducing weight structure as well as vectorisation techniques amenable to parallelised computation.

# References


Agakov, F. V. and Barber, D. An auxiliary variational method. In *International Conference on Neural Information Processing*, pp. 561–566. Springer, 2004.

Atanov, A., Ashukha, A., Struminsky, K., Vetrov, D., and Welling, M. The deep weight prior. In *International Conference on Learning Representations*, 2019.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. In *International Conference on International Conference on Machine Learning*, pp. 1613–1622, 2015.

Bradshaw, J., Matthews, A. G. d. G., and Ghahramani, Z. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Carlin, B. P. and Chib, S. Bayesian model choice via Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(3):473–484, 1995.

Daxberger, E., Nalisnick, E., Allingham, J. U., Antorán, J., and Hernández-Lobato, J. M. Expressive yet tractable bayesian deep learning via subnetwork inference. *arXiv preprint arXiv:2010.14689*, 2020.

Doucet, A. A note on efficient conditional simulation of Gaussian distributions. *Technical report, University of British Columbia*, 2010.

Dusenberry, M. W., Jerfel, G., Wen, Y., Ma, Y.-a., Snoek, J., Heller, K., Lakshminarayanan, B., and Tran, D. Efficient and scalable Bayesian neural nets with rank-1 factors. In *International Conference on Machine Learning*, pp. 9823–9833, 2020.

Foong, A. Y., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. 'in-between' uncertainty in Bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Ghosh, S., Yao, J., and Doshi-Velez, F. Model selection in Bayesian neural networks via horseshoe priors. *Journal of Machine Learning Research*, 20(182):1–46, 2019.

Graves, A. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330, 2017.

Gupta, A. K. and Nagar, D. K. *Matrix variate distributions*, volume 104. CRC Press, 2018.

Havasi, M., Peharz, R., and Hernández-Lobato, J. M. Minimal random code learning: Getting bits back from compressed model parameters. In *International Conference on Learning Representations*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pp. 630–645. Springer, 2016b.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Hoffman, M. D. and Gelman, A. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Hoffman, Y. and Ribak, E. Constrained realizations of Gaussian fields-a simple algorithm. *The Astrophysical Journal*, 380:L5–L8, 1991.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

Izmailov, P., Maddox, W. J., Kirichenko, P., Garipov, T., Vetrov, D., and Wilson, A. G. Subspace inference for Bayesian deep learning. In *Uncertainty in Artificial Intelligence*, 2019.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

Journel, A. G. and Huijbregts, C. J. *Mining geostatistics*. Academic press London, 1978.

Karaletsos, T. and Bui, T. D. Hierarchical Gaussian process priors for Bayesian neural network weights. *arXiv preprint arXiv:2002.04033*, 2020.

Kendall, A. and Gal, Y. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pp. 5574–5584, 2017.

Khan, M. E. E., Immer, A., Abedi, E., and Korzepa, M. Approximate inference turns deep networks into Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 3094–3104, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html*, 6:1, 2009.

Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A., and Courville, A. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

Lee, J., Sohl-dickstein, J., Pennington, J., Novak, R., Schoenholz, S., and Bahri, Y. Deep neural networks as Gaussian processes. In *International Conference on Learning Representations*, 2018.

Leibfried, F., Dutordoir, V., John, S., and Durrande, N. A tutorial on sparse gaussian processes and variational inference. *arXiv preprint arXiv:2012.13962*, 2020.

Louizos, C. and Welling, M. Multiplicative normalizing flows for variational Bayesian neural networks. In *International Conference on Machine Learning*, pp. 2218–2227, 2017.

Louizos, C., Ullrich, K., and Welling, M. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pp. 3288–3298, 2017.

Ma, C., Li, Y., and Hernández-Lobato, J. M. Variational implicit processes. In *International Conference on Machine Learning*, pp. 4222–4233, 2019.

MacKay, D. J. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 354(1):73–80, 1995.

Matthews, A. G. d. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

Mishkin, A., Kunstner, F., Nielsen, D., Schmidt, M., and Khan, M. E. Slang: Fast structured covariance approximations for Bayesian deep learning with natural gradient. In *Advances in Neural Information Processing Systems*, pp. 6245–6255, 2018.

Neal, R. M. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

Ober, S. W. and Aitchison, L. Global inducing point variational posteriors for Bayesian neural networks and deep Gaussian processes. *arXiv preprint arXiv:2005.08140*, 2020.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pp. 13991–14002, 2019.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. 2017.

Ranganath, R., Tran, D., and Blei, D. Hierarchical variational models. In *International Conference on Machine Learning*, pp. 324–333, 2016.

Ritter, H., Botev, A., and Barber, D. A scalable Laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

Salimans, T., Kingma, D., and Welling, M. Markov chain Monte Carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pp. 1218–1226, 2015.

Salimbeni, H. and Deisenroth, M. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 4588–4599, 2017.

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pp. 1257–1264, 2006.

Sun, S., Zhang, G., Shi, J., and Grosse, R. Functional variational Bayesian neural networks. In *International Conference on Learning Representations*, 2019.

Swiatkowski, J., Roth, K., Veeling, B. S., Tran, L., Dillon, J. V., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. The k-tied normal distribution: A compact parameterization of Gaussian mean field posteriors in Bayesian neural networks. In *International Conference on Machine Learning*, pp. 6631–6641, 2020.

Tanno, R., Worrall, D. E., Ghosh, A., Kaden, E., Sotiropoulos, S. N., Criminisi, A., and Alexander, D. C. Bayesian image quality transfer with cnns: exploring uncertainty in dmri super-resolution. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 611–619. Springer, 2017.

Tipping, M. E. and Bishop, C. M. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Titsias, M. and Lázaro-Gredilla, M. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning*, pp. 1971–1979, 2014.

Wen, Y., Tran, D., and Ba, J. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020.

Xu, H., Liu, B., Shu, L., and Yu, P. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, 2018a.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pp. 5852–5861, 2018b.

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical stochastic gradient MCMC for Bayesian deep learning. In *International Conference on Learning Representations*, 2020.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No] (Our work considers reducing model parameter sizes for Bayesian neural networks.)
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] (this is for the corrupted CIFAR dataset (Hendrycks & Dietterich, 2019).)
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]